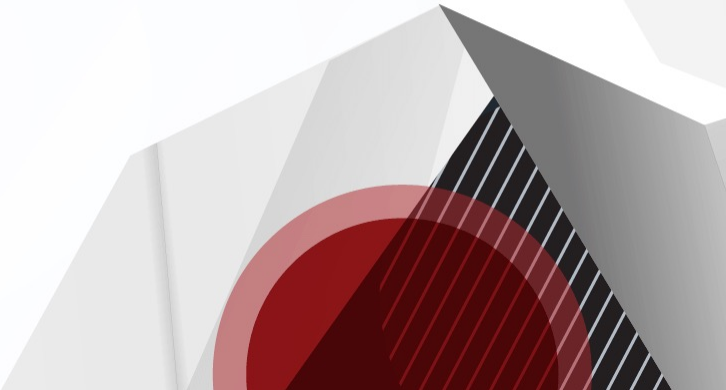
UMBC Malware Analysis Class

Christopher Gardner
Senior Reverse Engineer, FLARE Team



Chris Gardner

- Based in Denver, CO
- Senior Reverse Engineer at FireEye/Mandiant
  - FLARE Team
- Graduated UMBC CMSC ‘18
  - Former Cyberdawg
  - Former TA for this class
    - I was RJ’s TA ☺
- For fun
  - Rock Climbing, Skiing, other Colorado things
  - CTFs

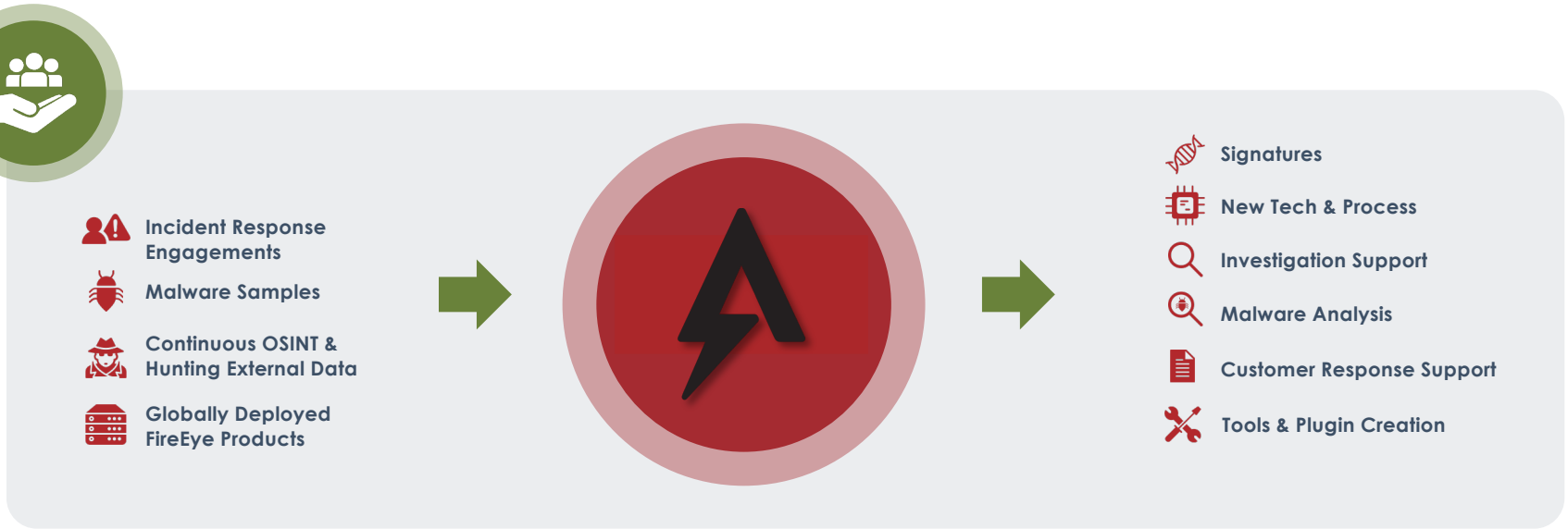
Agenda

- What is FLARE?
- What it's like to be a Malware Analyst
- A look at some cool FLARE tools
- Feature Presentation: "Beating the Malware Pinata"
- Internship/job pitch
- Q&A

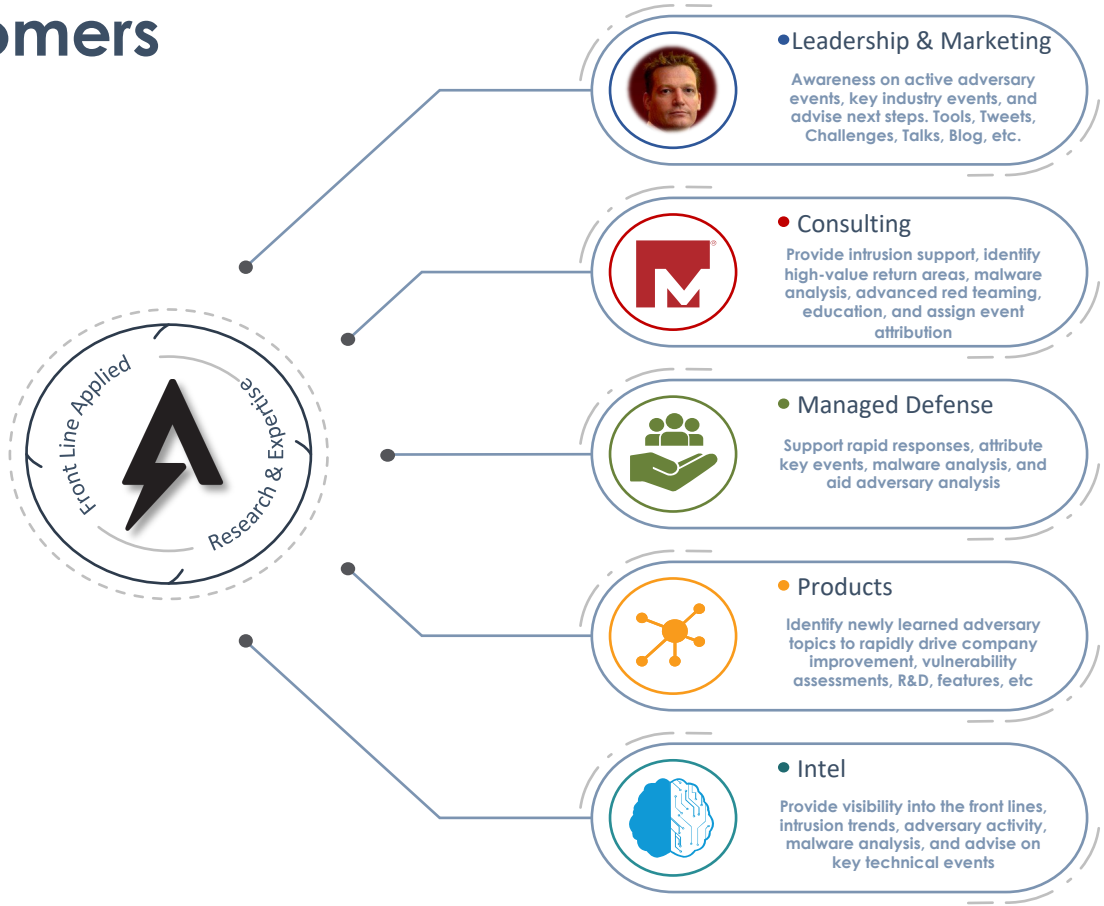
FRONT LINE APPLIED RESEARCH & EXPERTISE

- **Mission**
  - Find Evil & Expand Wisdom
- **Vision**
  - Discover, Enrich, and Broker Front-Line Knowledge to Internal and External customers

Finding Evil and Expanding Wisdom – From the Front Line

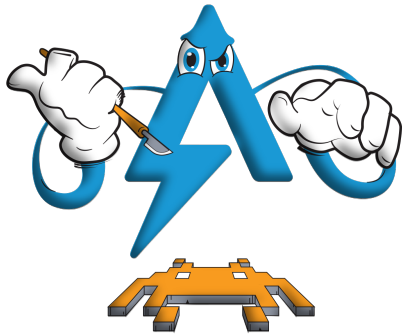


FLARE Customers



FLARE

- Elite team of reverse engineers and researchers
- International, remote team (~40 people)
- Reverse engineer pretty much all the malware at FireEye/Mandiant
  - Huge stream of interesting stuff to look at ☺
- Also find bugs sometimes
- Release cool tools (open source!)
- Teach classes on Malware Analysis
- Create/implement binary similarity tools



FLARE In the news

Zero-Day Exploits in SonicWall Email Security Lead to Enterprise Compromise

April 20, 2021 | by Josh Fleischer, Chris DiGiamo, Alex Pennino

Check Your Pulse: Suspected APT Actors Leverage Authentication Bypass Techniques and Pulse Secure Zero-Day

April 20, 2021 | by Dan Perez, Sarah Jones, Greg Wood, Stephen Eckels

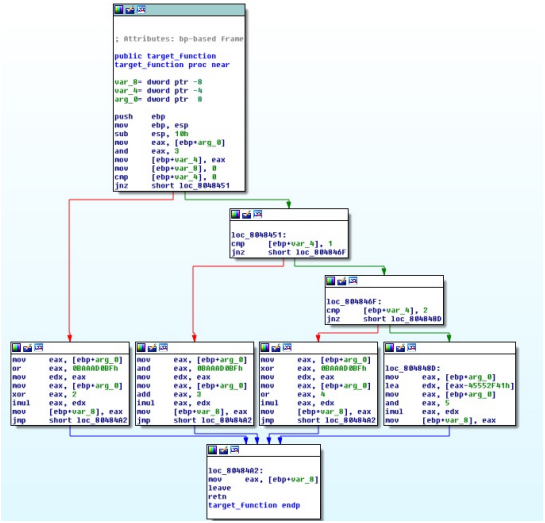
Highly Evasive Attacker Leverages SolarWinds Supply Chain to Compromise Multiple Global Victims With SUNBURST Backdoor

December 13, 2020 | by FireEye

Malware Operations

- Support the entire company in-depth RE and MA
  - From Incident Response for clients to Internal IT
  - Sample analysis, decoders, and specific questions answered
  - Reports contain detections, capabilities, and detailed analysis
- Malware "Queues" – Staffed with six analysts + Queue Ops
  - Mandiant Consulting
  - Mandiant Managed Defense
  - Mandiant Intel – "Hot List"
  - Mandiant Intel Analyst Access / I3 Support
  - FLARE Advanced Practices
- Mentoring Program
  - Develop new analysts, define analysis process, and oversight

2020 YTD Statistics
Analyzed Samples: 1452
Escalations: 48



Offensive Task Force (OTF)

- Elite group focused on offensive support of company and customers
  - Vulnerability/Exploit analysis
  - Zero-day reporting and coordination
  - Custom Tool Development
- Support the Red Team Function
  - Write malware for our red team
  - Application level assessments
- Application Security Assessments
  - Internal – Work closely with Product Security to break products before others
  - External – Team up with Mandiant consulting on low-level assessments
- So far in 2020
  - 15 Application Security Assessments for Customers, Product Security, and MD
  - Over 21 vulnerabilities reported across a multitude of vendors

External Education

- Offering training at conferences and client sites all over the world
  - Essentials of Malware Analysis (2 days)
  - Malware Analysis Crash Course (3 days)
  - Malware Analysis Master Class (5 days)
  - Customized Malware Analysis Course (2-10 days)
  - Router Backdoor Analysis Class (2 days)
  - MacOS Malware Analysis Crash Course (2 days)
- Pivoted to teach 31 courses online during Covid
- Development of new offerings
  - New Hotness: Malware Authoring and Repurposing

Applied Research

- Build tools to make automated analysis better, and augment manual analysis
- Given a malware sample
  - Is this similar to any other malware we know about?
  - Can we automatically unpack this sample?
  - What capabilities does this sample have?
  - What indicators can we automatically extract?

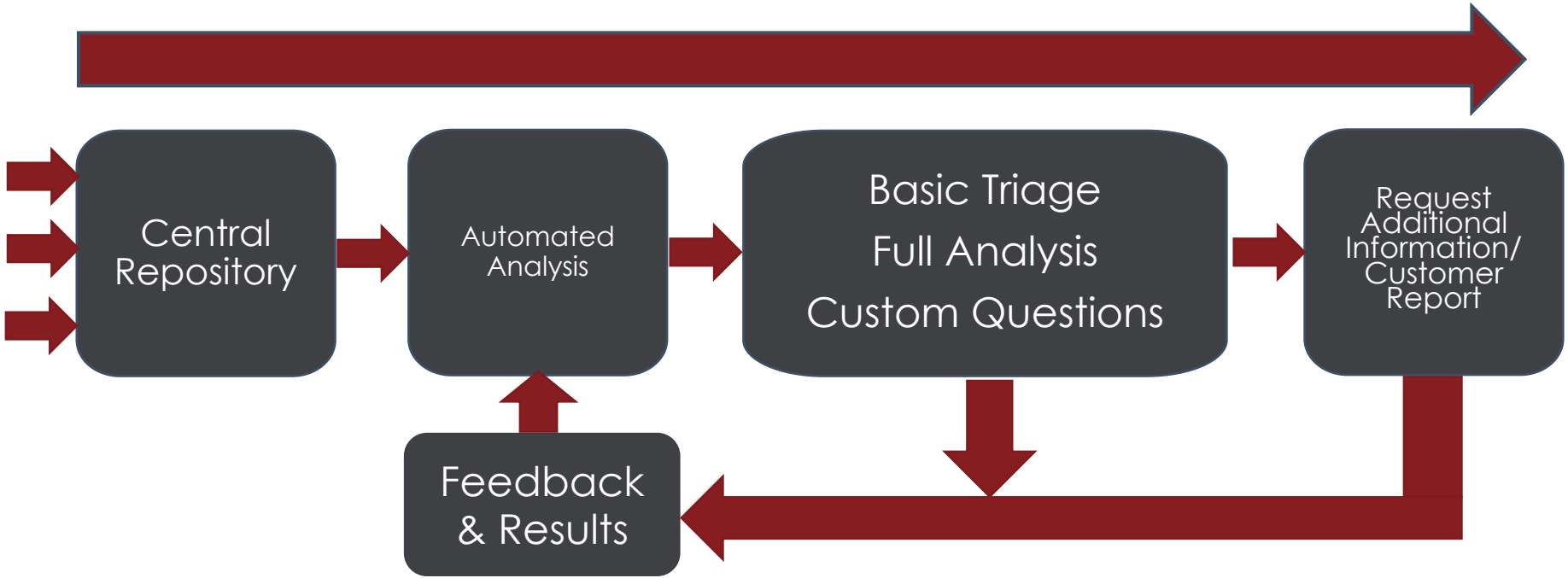
What does a Malware Analyst Do?

- Analyze malware, write reports
  - Reports are more freeform than your homework assignments
  - Sometimes there are special requests
- Make signatures for malware (sometimes)
- Do other research
  - Better malware detection strategies
  - Vulnerability research
  - Data science

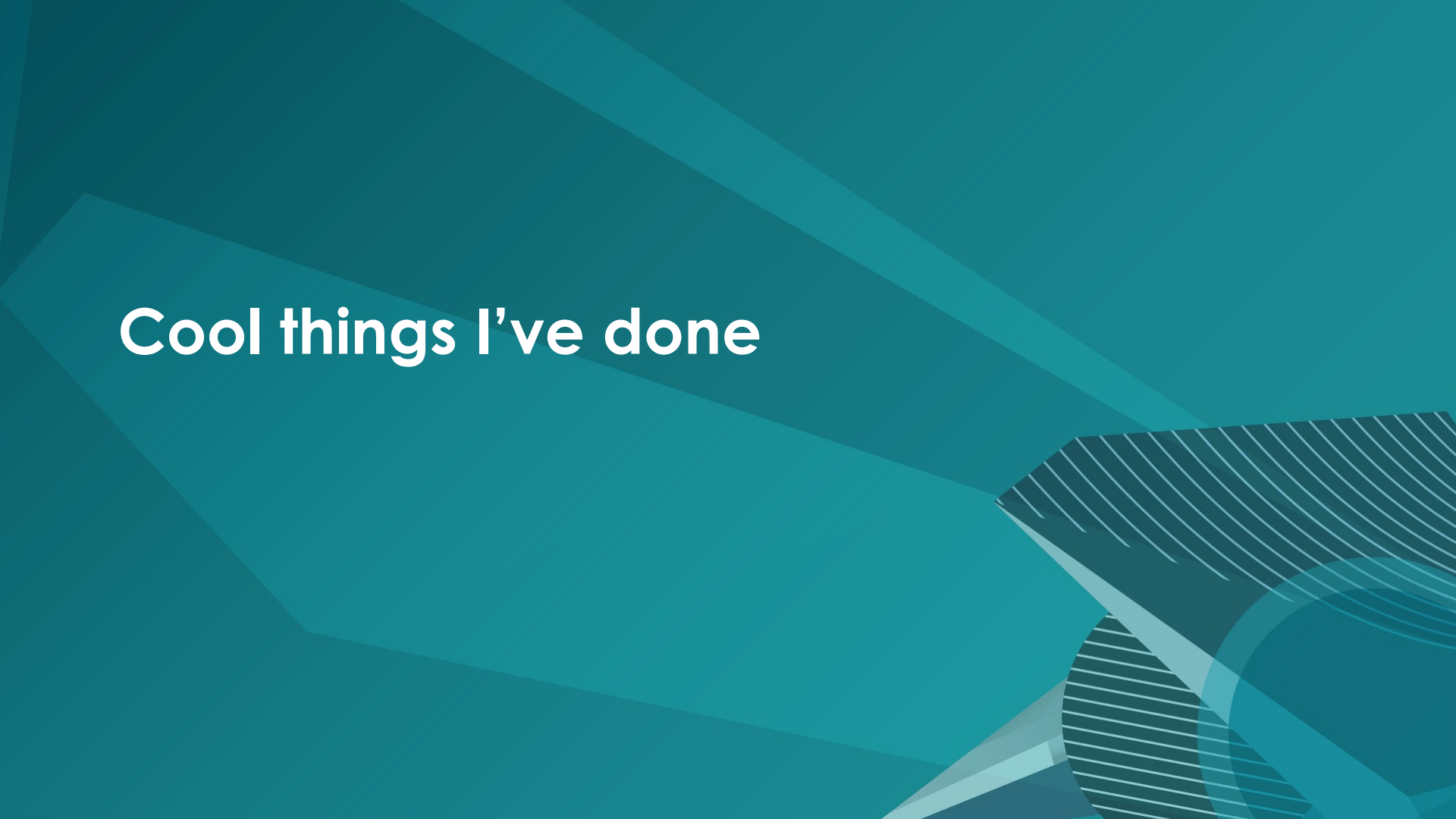
A day in the life of a Reverse Engineer

- Varies depending on what week it is
- Sometimes on the malware queue, neck deep in IDA Pro reversing cool stuff
- Sometimes taking two weeks to write an automated unpacker
- Writing/giving conference talks
- Teaching/developing courses
- Showing my coworkers some sick shellcode

Malware Operations Workflow – Horizon & JIRA Tickets



Cool things I've done



Teaching!

Research

Activation Analysis of a Byte-Based Deep Neural Network for Malware Classification

Scott E. Coull
FireEye, Inc.
scott.coull@fireeye.com

Christopher Gardner
FireEye, Inc.
christopher.gardner@fireeye.com

***Abstract*—Feature engineering is one of the most costly aspects of developing effective machine learning models, and that cost is even greater in specialized problem domains, like malware classification, where expert skills are necessary to identify useful features. Recent work, however, has shown that deep learning models can be used to automatically learn feature representations directly from the raw, unstructured bytes of the binaries themselves. In this paper, we explore what these models are learning about malware. To do so, we examine the learned features at multiple levels of resolution, from individual byte embeddings to end-to-end analysis of the model. At each step, we connect these byte-oriented activations to their original semantics through parsing and disassembly of the binary to arrive at human-understandable features. Through our results, we identify several interesting features learned by the model and their connection to manually-derived features typically used by traditional machine learning models. Additionally, we explore the impact of training data volume and regularization on the quality of the learned features and the efficacy of the classifiers, revealing the somewhat paradoxical insight that better generalization does not necessarily result in better performance for byte-based malware classifiers.**

I. Introduction

in the malware classification domain. In this paper, we seek to answer this question by providing a deep and broad analysis of activations in a byte-based deep neural network classifier that is representative of the architectures proposed in previous work. Unlike previous work, however, we expand our analysis beyond simply looking at the location of the activations to understanding the specific features that are learned and their connection to the semantics of the executable as a malware analyst would understand them. We perform this analysis under a variety of training regimes to gain a better understanding of the bias-variance tradeoff that exists for byte-based models in the unique problem area of malware classification.

Specifically, we examine the question at three levels: (1) the embedding layer to uncover learned similarities among independent byte values, (2) the first convolutional layer to identify low-level features over short byte sequences, and (3) end-to-end analysis for complex features combined over several layers of aggregation in the model. At each of these layers, we compare three models trained under increasing data volumes and levels of regularization to understand the

Special Projects

- Automated parsing and decryption of malware network traffic
- Reverse engineering medical devices for ‘compliance’
- Advanced sandbox sorcery
- Continuously scan the internet for new C2 servers

Skills needed for Malware Analysis

- Writing!
- Disassembly
- Debugging/Dynamic analysis
- Windows Internals
- Programming/scripting
- More advanced stuff
  - Cryptanalysis
  - Program analysis
  - Emulation

FLARE Public Tooling

FLARE VM

Windows VM with many malware analysis tools installed

FLOSS

Automatic deobfuscation of strings (sometimes)

Capa

Automatically detecting malware capabilities

FAKENET-NG

Internet simulation that actually works

flare-ida

Loads of IDA Pro plugins to automate common tasks

Speakeasy

Emulator designed to execute kernel & user space binaries & shellcode

FireMLaaS

Machine learning based malware classification (on VirusTotal)

And More!

FLARE-VM

- Build a Windows VM geared towards Malware Analysis
- Updated monthly
- All the tools FLARE uses on a regular basis
  - If we use something and it isn't in FLARE VM, we add it
- Free!
- Easy to install – just run a Powershell script
- Other flavors available as well
  - CommandoVM – red team focused toolkit

Quick tour of FLARE VM + other non-FLARE tools

- Sysinternals – procmon, autoruns, procexp, etc
- PE Tools – pestudio, CFF explorer, DIE, etc
- Lots of disassemblers/enhancers – IDA, Binary Ninja, IDR, dnSpy, jd-gui
- Hex editors: 010, HxD
- Debuggers: x64dbg, OllyDbg, WinDbg
- All the FLARE tools
- Python + helpful libs
- And so much more!

FLOSS

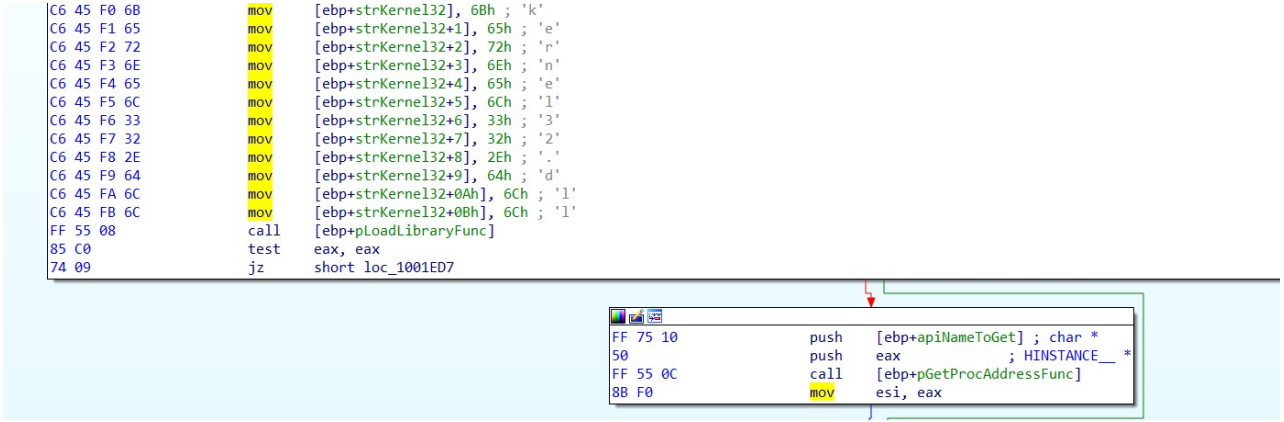
FLOSS – FireEye Labs Obfuscated String Solver

- More than just strings
- Automatically deobfuscates strings used by the binary
- Much quicker than manually pulling them out with a debugger
- Simple (floss my_program), but tuneable
- Will also pick up regular strings and stack strings

FLOSS internals

- How does FLOSS work?
- Uses Vivisect (Python program analysis library) under the hood
- Identify possible string decoding functions
- Extract arguments for those functions
- Emulate the functions
- Look for human readable strings in the memory output

Stack strings



FAKENET-NG

FAKENET-NG

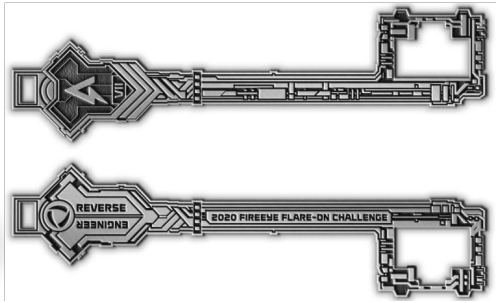
- Successor to Fakenet
- Replaces inetsim, ApateDNS, etc
- Super easy to set up (don't need to run 2 VMs like inetsim)
- Makes the malware think it can access the internet
- Has handlers for a variety of protocols/services
  - HTTP/S, SMTP, FTP, etc
- Can do TLS interception too!
- Everything is saved in a pcap for you to analyze later, as well as logged to the console

CAPA

- Automatically extract some capabilities from malware
- Uses CAPA rules – sorta like YARA but with more analysis
- Can reference specific assembly instructions, constants used in code, etc
  - Example: can identify if a specific constant in AES is used by the sample
- Open source, and open source rules that anyone can contribute to:
  - https://github.com/fireeye/capa
  - https://github.com/fireeye/capa-rules

The FLARE On Challenge

- Multiple binary CTFs based around reverse engineering
- **5,648 registered participants in 2020**
  - 260 winners completed
  - 3,574 completed at least one challenge, record!
- All past challenges www.FLARE-On.com with solutions and on FireEye blog
- Diverse puzzles
  - Nintendo, Android, Virtualization, Steg, .NET, etc
- Prize & Bragging Rights
- Largest RE Competition in the World



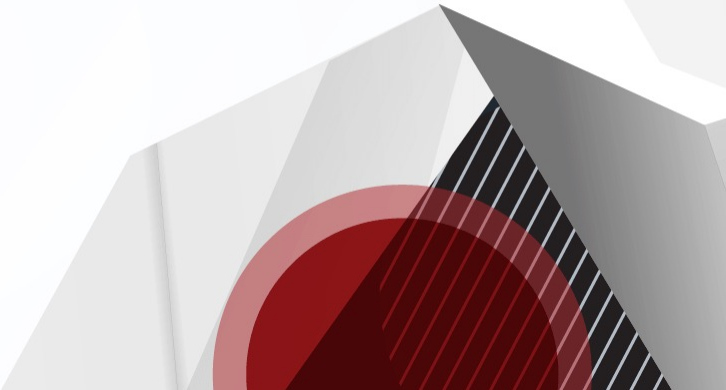
Sharing With The Community

- Get our **code** - https://github.com/fireeye/
- Read our **blog** - http://www.fireeye.com/blog with tag "FLARE"
- Read our **whitepapers** – M-Trends, Synful Knock, WMI, etc
- Compete in our **challenge** – http://www.flare-on.com/
- Play with our free **tools** - https://www.fireeye.com/services/freeware.html

Beating the Malware Piñata

Malware of the Month

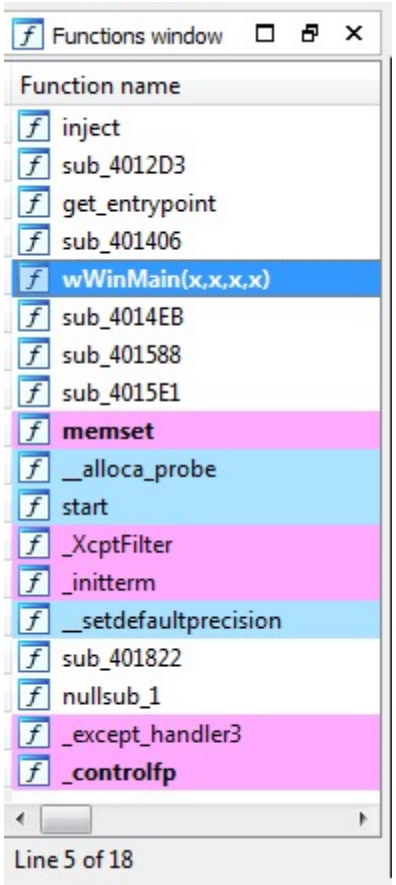
Chris Gardner
Reverse Engineer



One week on the Intel queue…

A very productive week, until now

- Perusing the intel queue, looking for my next ticket
- `0178a69c43d4c57d401bf9596299ea57`, submitted by our Threat Intel team
- "Potential LOCKLOAD? Would be very interested to know if there are links to Fallout Team"
  - Family names, hooray!
  - Has a TIS ticket attached!
  - Wait, 2 MB?
    - Not Go
    - Not Delphi
    - C++
    - Uh oh!



2 MB, 18 functions… hm…

Nurse, get me 20 CCs of explorer.exe, STAT!

- Malware reads shellcode from the copy of itself on disk
- Creates `explorer.exe` in suspended mode
- Writes shellcode to suspended process using WriteProcessMemory()
- Manually resolve functions to create an import table
- Resume thread
- ????
- Profit!

2 ways to solve

Static way

- Lots of work
- Works every time, no need to redo anything on later samples
- Create a Binary Ninja loader plugin that patches in the import table
- Cool, technical, slick

Dynamic way

- Moderate amount of work
- Have to redo completely each time
- Run the process, dump memory
- Brash, uncultured, dirty

2 ways to solve

Static way

- Lots of work
- Works every time, no need to redo anything on later samples
- Create a Binary Ninja loader plugin that patches in the import table
- Cool, technical, slick

Dynamic way

- Moderate amount of work
- Have to redo completely each time
- Run the process, dump memory
- Brash, uncultured, dirty

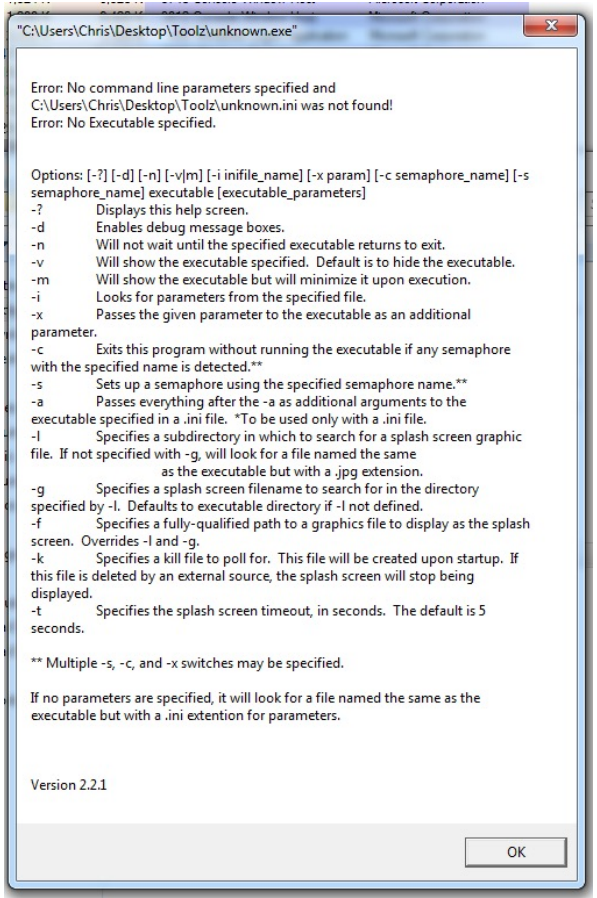
Some cool shellcode

- Alright! A juicy payload ripe with indicators
- Oh wait:
  - Generate temporary filename starting with @AE
  - Read malware on disk, decode some more code
  - Write to temporary file and execute
- *It's just a dropper* ☹

WHACK!

93D1BABAE7EAD19B4551DBFA57E858CE

- Rewrites original file with a legit IBM utility
- Takes a very long time just to show an error
- 🤔



280200E5C0F57EBC01662C6B9976B7D9 - @AE1.tmp.exe

- Nurse! More `explorer.exe`!
- Here we go again
- Of course this is the last time, so let's just do the dynamic way again ☺
  - It was at this moment that Chris sealed his fate, as this was not the last time, nor the second to last time


?
Pokémon


?!
Pokémon

WHACK

times two!

081BFF47D9069448A9AF0DACD064469E – dll_suspender.dll

- Library that is loaded in memory by the second dropper
- Contains juicy persistence indicators ☺
- Implemented as an annoying to reverse COM object for some reason
  - No other obfuscation of indicators
- Saves a copy of `ws2_32.dll` (Winsock) for some reason

6cb9e6476ca972812c1c80bd68e031d1 – WdExt.exe

- At last, the main dropper!
  - But not the final one
- Drops **8** PEs
- Injects into explorer.exe again

Created files

- Filename: `mydll.dll`
  - Path: `%TEMP%`
  - MD5: `<varies>`
- Filename: `arc.dll`
  - Path: `%APPDATA%\Microsoft\Identities\<username>`
  - MD5: `2D9DF706D1857434FCAA014DF70D1C66`
- Filename: `arc.dll`
  - Path: `%APPDATA%\Microsoft\Identities\<username>`
  - MD5: `2D9DF706D1857434FCAA014DF70D1C66`
- Filename: `att.dll`
  - Path: `%APPDATA%\Microsoft\Windows\Addins`
  - MD5: `FFFA05401511AD2A89283C52D0C86472`
- Filename: `dis.dll`
  - Path: `%APPDATA%\Microsoft\Common\Shared`
  - MD5: `1FCC5B3ED6BC76D70CFA49D051E0DFF6`
- Filename: `fil.dll`
  - Path: `%APPDATA%\Microsoft\Shared\Modules`
  - MD5: `D0C9ADA173DA923EFABB53D5A9B28D54`
- Filename: `sha.dll`
  - Path: `%APPDATA%\Microsoft\Repairs`
  - MD5: `6A9461F260EBB2556B8AE1D0BA93858A`
- Filename: `usd.dll`
  - Path: `%APPDATA%\Microsoft\Caches\Files`
  - MD5: `F1C9F4A1F92588AEB82BE5D2D4C2C730`
- Filename: `launch.exe`
  - Path: `%APPDATA%\Microsoft\Defender`
  - MD5: `DAAC1781C9D22F5743ADE0CB41FEAEBF`
- Filename: `wtmps.exe`
  - Path: `%TEMP%`
  - MD5: `75c1467042b38332d1ea0298f29fb592`

WHACK

times eight!

The libraries

- Drops 6 DLLs that serve as libraries for the malware

Name	Internal Name	Description
dis.dll	dll_diskscan.dll	Scans all drives on the system, passes to arc.dll and fil.dll
arc.dll	dll_archive.dll	ZLIB wrapper
att.dll	dll_attachdetect.dll	Hides files used by the malware from Save file or Open file dialogs
fil.dll	dll_fileinfect.dll	Infects PE files with the malware, also compresses files for exfiltration
sha.dll	dll_netsharedetect.dll	Scans network shares for use with dis.dll
usd.dll	dll_usbdetect.dll	Scans USB drives for use with dis.dll

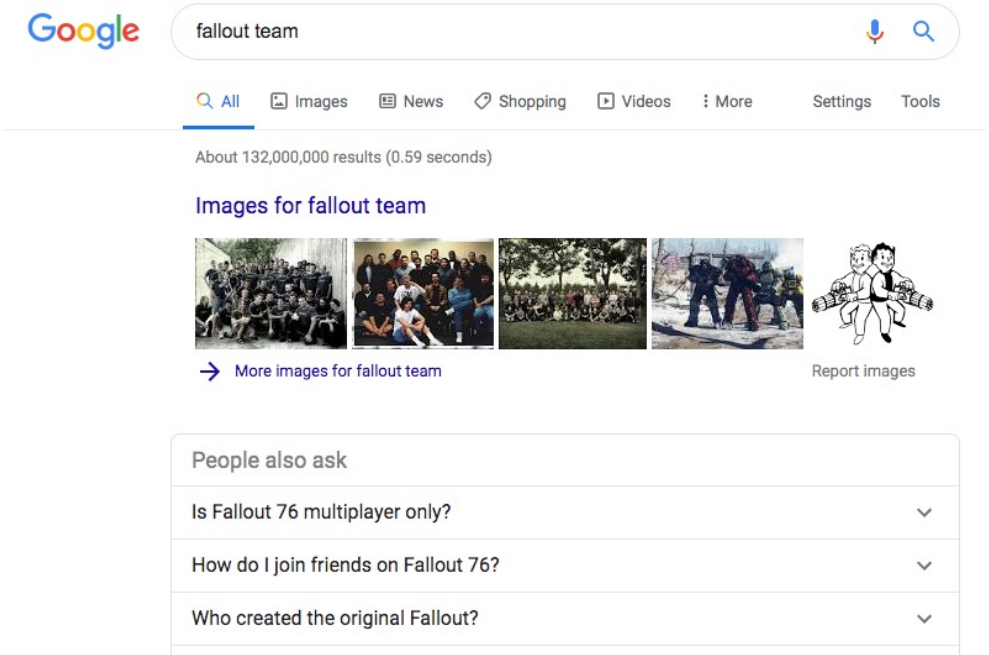
A break – time for some research

- *"Potential **LOCKLOAD**? Would be very interested to know if there are links to Fallout Team"*
- Quickly ruled out LOCKLOAD
- Is it Fallout team?

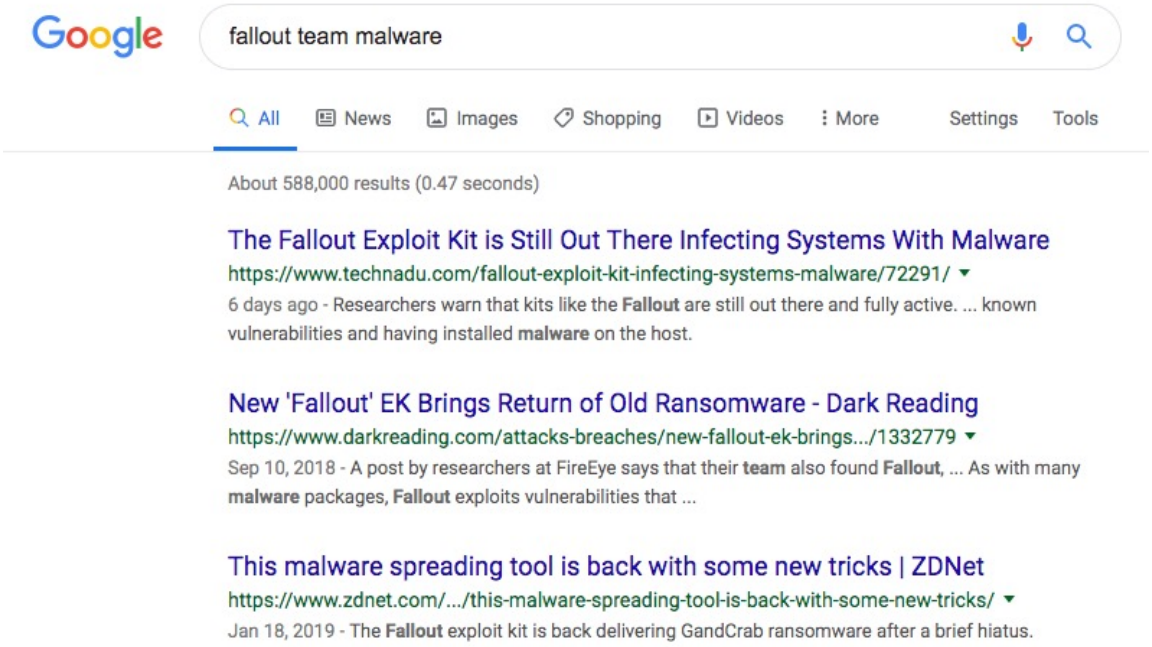
A break – time for some research

- *"Potential **LOCKLOAD**? Would be very interested to know if there are links to Fallout Team"*
- Quickly ruled out **LOCKLOAD**
- Is it Fallout team?

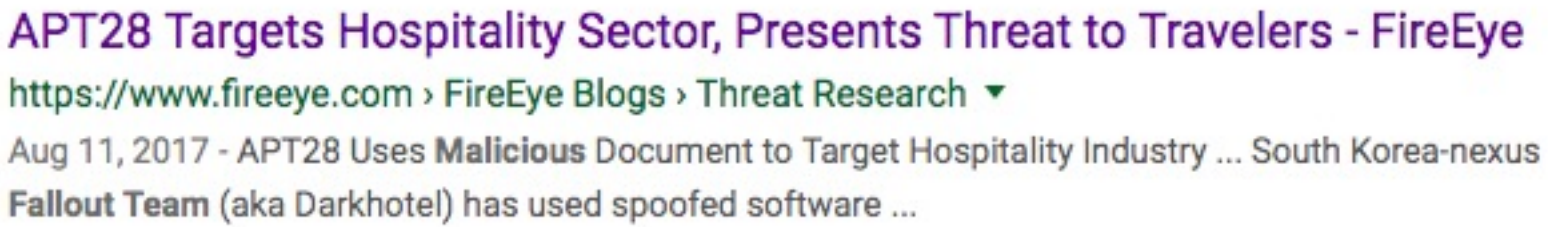
- Google is unhelpful



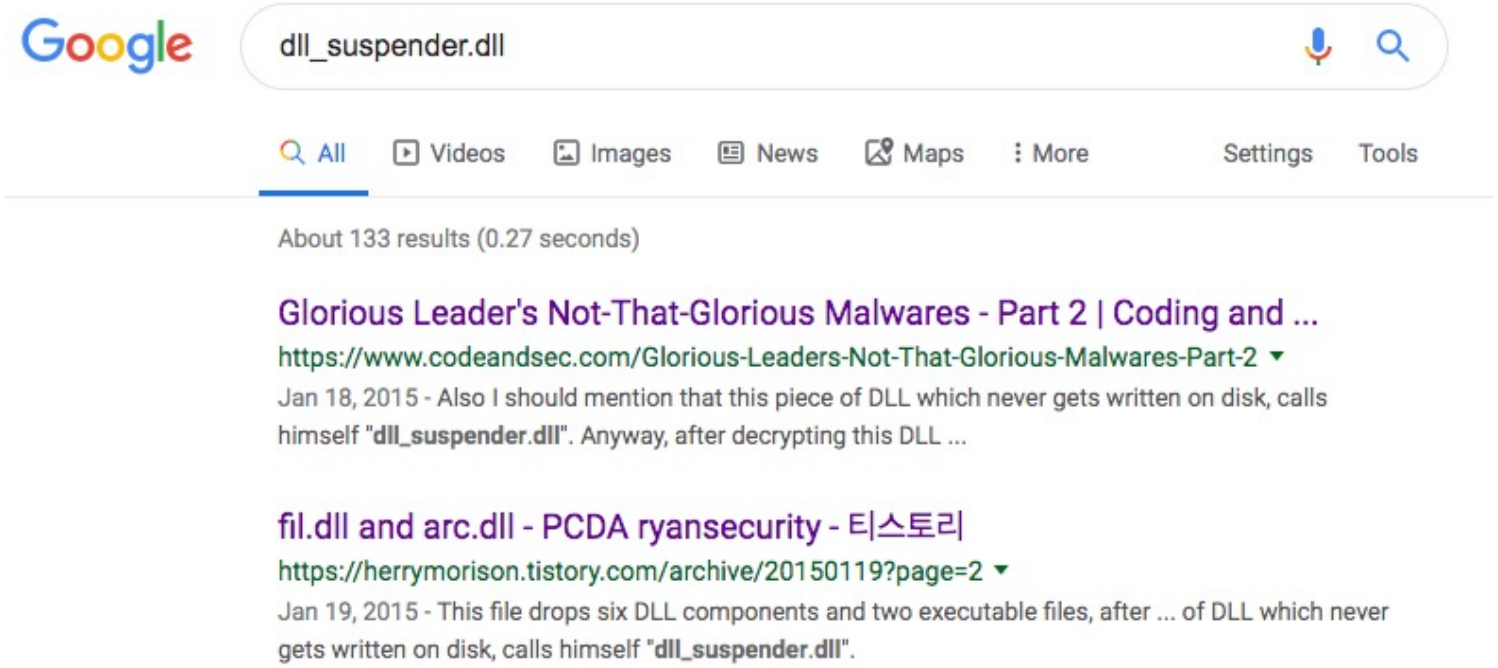
Fallout Exploit kit != Fallout



thank u FireEye



What about those internal DLL names? Maybe someone has analyzed this before…



`DAAC1781C9D22F5743ADE0CB41FEAEBF - launch.exe`

- One of the two EXEs dropped by the main dropper.
- Analyzed by the blog post! Yeah! ☺
- Injects into explorer, drops persistence, loads the libraries into `explorer.exe` processes
- ezpz

75C1467042B38332D1EA0298F29FB592 – wtmps.exe

- Not mentioned in the blog post at all
  - Have to get my hands dirty again
- Quite different from all the other malware in this chain
  - Doesn't inject into explorer, just runs everything itself
- …but still just another dropper

WHACK

78D3C8705F8BAF7D34E6A6737D1CFA18 – mscaps.exe

- Drops persistence ☺
  - wooo, indicators!
- This one isn't a dropper!
- ….it's a launcher

WHACK

978888892A1ED13E94D2FCB832A2A6B5 - wtime32.dll

- This is it folks
- The final payload
- The final frontier
- The only thing standing between me and the sweet sweet feeling of *closing 8 tickets in 1 minute*
  - stats=padded

wtime32.dll

- Basic backdoor over DNS
- Has some commands
- Wait, what's that **ts** command? Oh no…

The malware supports a few different commands:

Name	Description
go	Keep alive
ti	Set sleep interval (in minutes)
dl	Inject a DLL on disk into a process
du	Inject shellcode into a process
ts	Connect to a secondary C2 host, over a custom binary protocol. See below. The port is also given by the C2 server.
de	Checksum a file
ex	Execute a command
un	Uninstall the malware and exit

WHACK

TWO C2 PROTOCOLS????

- The payload also includes a more powerful custom binary protocol that it can use
- The Kaspersky report on DarkHotel missed this ☺

Name	Description
_prc	Get loaded modules in all processes (includes the process name as a byproduct, so acts as a listing of processes as well)
_quit	Exit this thread of communication
_dir	Send a directory listing (includes some extended info)
_del	Delete a file
_exe	Execute a command (no output is sent back to the C2 server)
_get	Send a file to the C2 server
_got	Send a file to the C2 server, and delete it afterwards (Kaspersky report is wrong)
_put	Recieve a file from the C2 server, set the timestamp to the timestamp of `gdi32.dll`
_dll	Inject a DLL on disk into a process
_dlu	NOP. Potentially a placeholder.
_cap	Take a screenshot, send to C2
_inf	Mine info from the system - time, Windows version, username, computer name, IP, `%SystemDrive%`, other network and disk info, all installed programs,
_cmd	Execute a command, send output back to C2 server

In the end….

- 1 analyst
- 8 tickets completed
- 16 total PE files dropped
- 15 PEs analyzed
- 2 incorrect/incomplete open source reports
- 45 hours of work over 1 month
- 1 box of kleenex used

Questions?

FLARE Jobs/internships

- Bit late for internships – but we are pretty much always hiring
  - https://www.fireeye.com/company/jobs.html
- FLARE internships work on real, important projects with real FLARE team members
  - Sandbox development
  - Binary similarity
  - Automated unpacking
- Much more out there than just FLARE, if it's infosec FireEye probably has an internship/job for it

Questions?

christopher.gardner@mandiant.com